Subgraph2vec algorithm

Thursday, 30th May,2019

Deepti Saravanan

REFERENCE.

subgraph2vec: Learning Distributed Representations of Rooted Sub-graphs from Large Graphs by A. Narayanan, M. Chandramohan, L. Chen, Y. Liu, and S. Saminathan.

Context

- Problem Statement
- Existing graph kernels - limitations
- Related Work
- Approach
- Evaluation

Problem Statement

To learning Distributed Representations of Rooted Sub-graphs from Large Graphs using graph2vec algorithm (an unsupervised representation learning technique to learn latent representations of rooted subgraphs present in large graphs.)

That is, for a set of graphs G = {g1,g2,...} and a given D, a vocabulary for all the rooted nodes around every node for every graph is to be extracted which includes the neighbours upto the mentioned degree(0 <= d <= D) such that sg_{vocab} = {sg1,sg2,..}.

Existing graph kernels - limitations

- ***Substructure Similarity:***

The proposed Weisfeiler-Lehman (WL) kernel ignores the initial subgraph similarities and considers each and every subgraph as a new feature, though they can be derived from the existing subgraphs by mere addition of one node or edge.

- ***Diagonal Dominance:***

The above limitation leads to increased dimensionality of the feature space, resulting in very few subgraphs common across various graphs. This increases the diagonal dominance, that is, the given graph is similar to just itself and not with any other, leading to poor classification.

Related Work

Deep Graph Kernels - by Yanardag and Vishwanathan : Existing solution.

Similarities between subgraphs capturing using the formulation:

$$K(G,G0) = \Phi(G)^T M \Phi(G0)$$

Where,

M is a |V|×|V| positive semi-definite matrix that encodes the relationship between substructures (using edit distance).

V represents the vocabulary of substructures obtained from the training data.

Related Work (contd.)

Calculation of Kernel Matrix: (i,j) value represents the similarity between G_i and G_j.

- Introduced the notion of “context window” between subgraphs.
- Substructures that co-occur in the same context tend to have high similarity.
- In the case of rooted subgraphs, all the subgraphs that encompass same degree of neighbourhood around the root node are considered as co-occurring in the same context.
- Objective was designed to make the embeddings of substructures that occur in the same context similar to one another. Thus, defining correct context directly affected the quality of embeddings.

Related Work (contd.) - 3 Assumptions

Assumption1:

- Only rooted subgraphs of same degree are considered as co-occurring in the same context.

  (But consider two nodes with degree=1 neighbours but different functionalities!).

Related Work (contd.)

Assumption2:

- Any two rooted subgraphs of different degrees never co-occur in the same context.

  (But consider two rooted nodes with different degrees but same functionalities!).

Related Work (contd.)

Assumption3:

- Every subgraph in any given graph has exactly same number of subgraphs in its context.

  (Violates the topological neighbourhood structure in graphs).

Approach

- Subgraph2vec considers all the rooted subgraphs (up to a certain degree) of neighbours of r as the context of $sg^{(d)}$ r, for all r in G with root r.
- Skipgram modification - considers radial context instead of fixed linear context.

Skipgram model - Considers the neighbouring subgraphs (can be bigram ot trigram) to influence the context of the current subgraph.

Radial context - For a neighbouring subgraph of degree d, we consider the subgraphs at (d-1) and (d+1) degrees to belong to the same context as that of the d^{th} degree subgraph.

Skip Gram Model

- Maximizes co-occurrence probability among the words that appear within a given context window.
- Given a context window of size c and the target word wt, the model attempts to predict the words that appear in the context of the target word, (wt−c,...,wt−c).
- Objective is to maximise the likelihood probability.

$$\sum_{t=1}^{T} log\ Pr(w_{t-c}, ..., w_{t+c}|w_t) \tag{5}$$

where the probability $Pr(w_{t-c}, ..., w_{t+c})$ is computed as

$$\Pi_{-c \leq j \leq c, j \neq 0} Pr(w_{t+j}|w_t) \tag{6}$$

Here, the contextual words and the current word are assumed to be independent. Furthermore, $Pr(w_{t+j}|w_t)$ is defined as:

$$\frac{exp(\Phi_{w_t}^T \Phi'_{w_{t+j}})}{\sum_{w=1}^{\mathcal{V}} exp(\Phi_{w_t}^T \Phi'_w)} \tag{7}$$

where Φ_w and $\Phi_w{}'$ are the input and output vectors of word w.

Negative Sampling

- Selects the words that are not in context at random instead of selecting all the words in the vocabulary.

- That is, if the word w appears in the context of another word w’, then the vector embedding of w is similar to that of w’.

Subgraph2vec Algorithm

- Considers all the neighbourhoods of rooted subgraphs around every rooted subgraph (up to a certain degree) as its corpus, and set of all rooted subgraphs around every node in every graph as its vocabulary.

2 STEPS:

1. Procedure to generate rooted sugraph around every node in a given graph.
2. Procedure to learn embeddings of those subgraphs.

Subgraph2vec Algorithm - Extracting rooted subgraphs

Computed for every node.

Inputs - Vertex v, Graph g and Degree d.

- Get all the breadth-first neighbours of the node v in N_v.
- For every neighbouring node v', we get its (d-1) degree subgraphs and store in $M_v^{(d)}$.
- We get the (d-1) subgraphs of the root node v and concatenate them with the sorted list $M_v^{(d)}$ to get the final intended subgraph sg_v.

Output - Root Subgraphs for every node, sg_v.

Algorithm

Algorithm 2: GETWLSUBGRAPH (v, G, d)

input : v: Node which is the root of the subgraph
$G = (V, E, \lambda)$: Graph from which subgraph has to be extracted
d: Degree of neighbours to be considered for extracting subgraph

output: $sg_v^{(d)}$: rooted subgraph of degree d around node v

begin
  $sg_v^{(d)} = \{\}$
  if $d = 0$ **then**
    $sg_v^{(d)} := \lambda(v)$
  else
    $\mathcal{N}_v := \{v' \mid (v, v') \in E\}$
    $M_v^{(d)} := \{\text{GETWLSUBGRAPH}(v', G, d-1) \mid v' \in \mathcal{N}_v\}$
    $sg_v^{(d)} := sg_v^{(d)} \cup \text{GETWLSUBGRAPH}$
    $(v, G, d-1) \oplus sort(M_v^{(d)})$
  return $sg_v^{(d)}$

Subgraph2vec Algorithm - Radial Skipgram

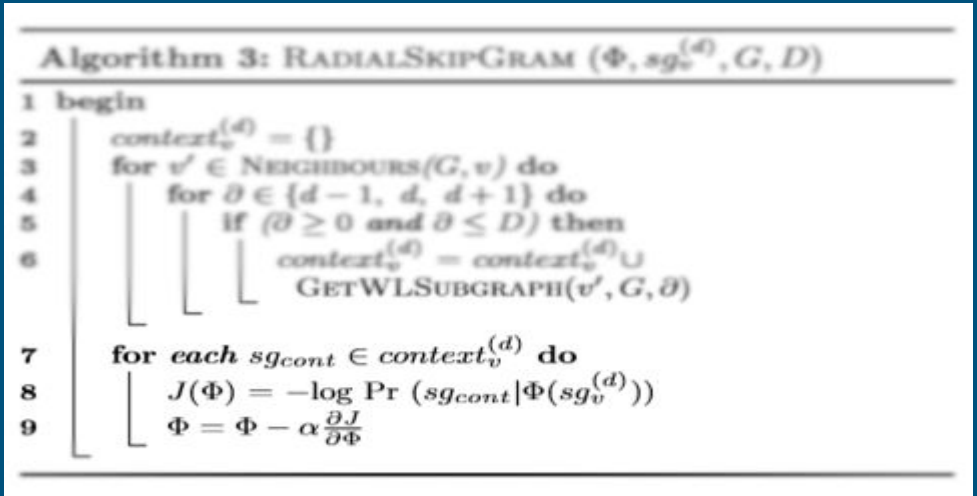
- Radial instead of linear context(Vanilla).
- Subgraphs of (d-1), d and (d+1) degrees are in the context of subgraph of degree d.

Algorithm 3: RADIALSKIPGRAM $(\Phi, sg_v^{(d)}, G, D)$

```
1 begin
2     context_v^(d) = {}
3     for v' ∈ NEIGHBOURS(G, v) do
4         for ∂ ∈ {d − 1, d, d + 1} do
5             if (∂ ≥ 0 and ∂ ≤ D) then
6                 context_v^(d) = context_v^(d) ∪
                    GETWLSUBGRAPH(v', G, ∂)
7     for each sg_cont ∈ context_v^(d) do
8         J(Φ) = −log Pr (sg_cont | Φ(sg_v^(d)))
9         Φ = Φ − α ∂J/∂Φ
```

Radial skipgram - embedding

- Given the current representation of target sub- graph $\Phi(sg_v^{(d)})$, we would like to maximize the probability of every subgraph in its context sg_{cont}.
- This may lead to huge number of labels equal to $|sg_{vocab}|$.
- Thus, probability distribution is approximated using negative sampling.



Negative sampling - Radial skipgram

- For every training cycle, a fixed number of subgraphs are selected called *negsamples*, whose embeddings will be updated during each iteration.

- Negative samples are subject to the following conditions:

1. If negsamples = {sgneg1,sgneg2,...}, then negsamples $\subset$ SGvocab.
2. |negsamples| << |SGvocab|.
3. negsamples $\cap$ context$^{(d)}$ = {}.

- This makes $\Phi(sg_v^{(d)})$ closer to the subgraphs in its context and also distances it away from the fixed number of negative samples.

Optimization

- Stochastic Gradient Descent(SGD) Optimizer is used to optimize the parameters.
- Backpropagation algorithm is incorporated.
- Learning rate alpha is empirically tuned.

Evaluation

Out-performed the Deep-WL Kernel in both accuracy (by 21%) and efficiency(pre-trained duration).

Where pre-training is the process of finding subgraph similarities.

THANK YOU